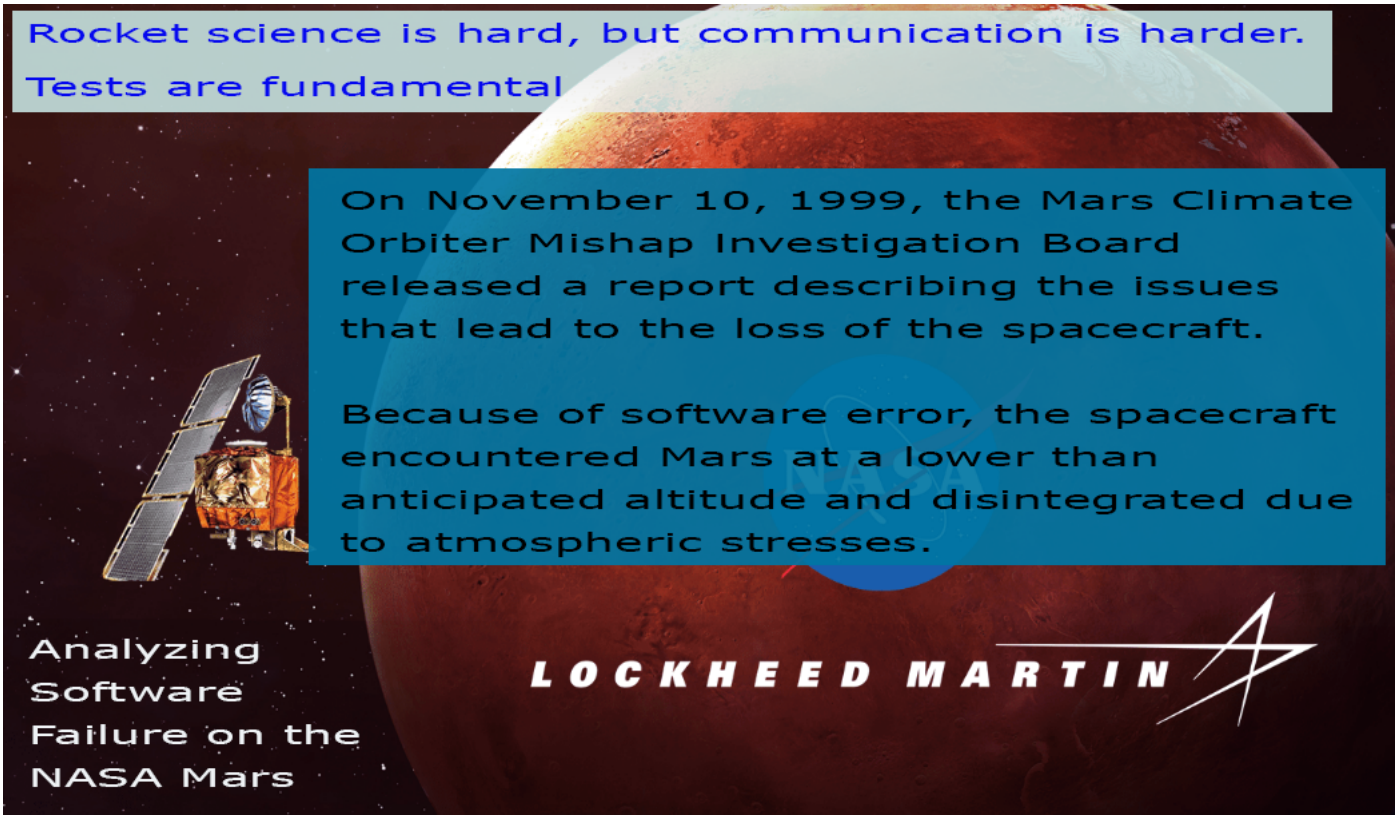
Software Quality Assurance Testing MAD5234 2020F

Peter Sigurdson

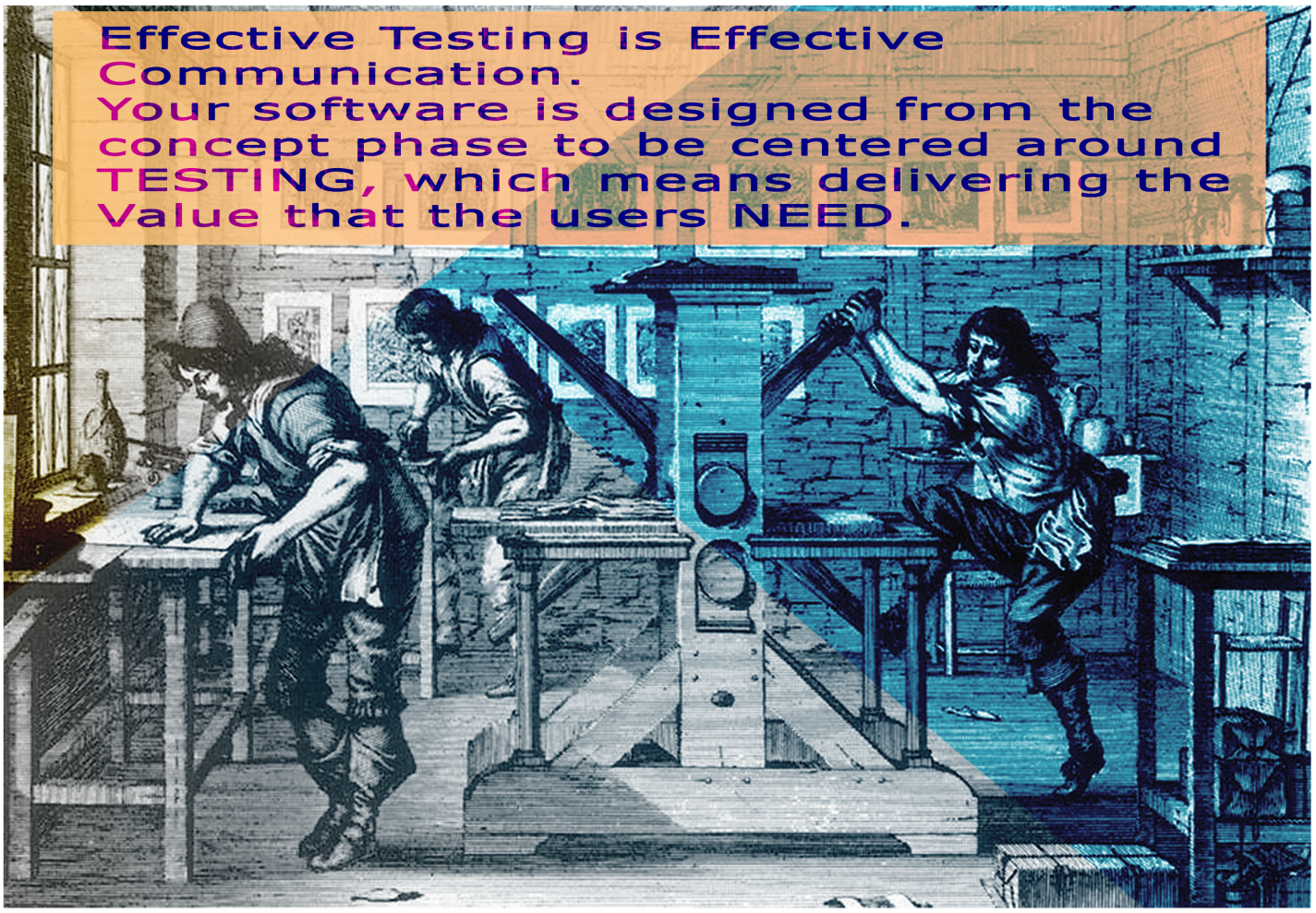
September 2020



https://t2m.io/MAD5234CourseVisualRoadmap

Evaluation Schedule

- ⋆ Sept 22 Test 1 20%
- ⋆ Sept 30 Test 2 20%
- ⋆ Sept 21 Assignment 1 15%
- ⋆ Sept 28 Assignment 2 15%
- ⋆ Sept 15 Activity 1 6%
- ⋆ Sept 18 Activity 2 6%
- ⋆ Sept 22 Activity 3 6%
- ⋆ Sept 25 Activity 4 6%
- ⋆ Sept 29 Activity 5 6%



Effective Testing is woven into the structure of the software development project from the beginning. You can't just bolt it on at the end after everything else is done. Effective Testing IS Effective Communication. What does the client NEED to make thier business work?

1 Day 02 Learning Outcomes:

2.1 Discuss the foundations of proper software properties, specifications and reliability versus safety. 2.2 Analyze and compare examples of program verification and correctness.

2 Topic 1: Case Study

Theory background on proving program correctness

3 Lecture Topics

4 Activity 1 6% : Building the IT System with Unified Process:

5 Outcome: Understand how to use the Tracability Matrix as the basis for our test cases.

5.1 Because we can't do effective testing until we have a good structural basis for writing well-formed test cases.

- ⋆
- ⋆ 1. Evaluate software testing and quality assurance techniques as part of an integrated discipline of software quality verification and validation.
- ⋆ 1.1 Discuss the discipline of software engineering.
- ⋆ 1.2 Discuss the importance of software quality attributes.
- ⋆ 1.3 Analyze the software testing lifecycle.

Introduction

This course is an introduction to the principles of software quality assurance.
The course addresses the concepts and practices of a software quality assurance function, as well as those aspects of:

- ⋆ project management,
- ⋆ software design,
- ⋆ and testing and configuration management,

as applicable to the development of quality software products.

Practices of Software Quality Assurance

- ⋆ Project Management for Software Quality Assurance
- ⋆ Software Design
- ⋆ Testing and Configuration Management

Course Learning Outcomes/Course Objectives

- ⋆ 1. Evaluate software testing and quality assurance techniques as part of an integrated discipline of software quality verification and validation.
- ⋆ 1.1 Discuss the discipline of software engineering
- ⋆ 1.2 Discuss the importance of software quality attributes.

⋆ 1.3 Analyze the software testing lifecycle.

⋆ 2. Assess software foundations, program correctness and verification and various failures, errors and faults and methods of software testing taxonomy.

⋆ 2.1 Discuss the foundations of proper software properties, specifications and reliability versus safety.

⋆ 2.2 Analyze and compare examples of program verification and correctness.

⋆ 2.3 Evaluate and determine courses of action taken from examples of failures, errors and faults.

⋆ 2.4 Analyze methods related to software testing taxonomy.

⋆ 3. Evaluate test generation concepts using functional and structural criteria.

⋆ 3.1 Discuss test generation concepts.

⋆ 3.2 Analyze examples of functional criteria.

⋆ 3.3 Analyze examples of structural criteria.

⋆ 4. Evaluate specifications of drivers and industry standards such as Oracle.

⋆ 4.1 Test oracle design specifications.

⋆ 4.2 Test driver design specifications.

⋆ 4.3 Test outcome analysis

⋆ 5. Discuss and evaluate the management of software testing.

⋆ 5.1 Discuss and analyze examples of metrics for software testing.

⋆ 5.2 Use software testing tools.

⋆ 5.3 Analyze and test product lines.

Teaching Day Plan

Day 01 MON Sept 14

1. Evaluate software testing and quality assurance techniques as part of an integrated discipline of software quality verification a
1.1 Discuss the discipline of software engineering.
1.2 Discuss the importance of software quality attributes.
1.3 Analyze the software testing life-cycle.

Day 02 TUE Sept 15

2.1 Discuss the foundations of proper software properties, specifications and reliability versus safety.
2.2 Analyze and compare examples of program verification and correctness.
Activity 1 6%

Day 03 WED Sept 16

2.3 Evaluate and determine courses of action taken from examples of failures, errors and faults.

Day 04 THU Sept 17

Review and reinforcement of topics covered so far: In class Review Activity
2.4 Analyze methods related to software testing taxonomy.

Day 05 FRI Sept 18

3.1 Discuss test generation concepts.
Activity 2 6%

Day 06 MON Sept 21

3.2 Analyze examples of functional criteria.
Assignment 1 15%

Day 07 TUE Sept 22

3.3 Analyze examples of structural criteria.
Activity 3 6%
Test 1 20%

Day 08 WED Sept 23

4.1 Test oracle design specifications.
4.2 Test driver design specifications.

Day 09 THU Sept 24

4.3 Test outcome analysis.

Day 10 FRI Sept 25

5.1 Discuss and anlayze examples of metrics for software testing.
Activity 4 6%

Day 11 MON Sept 28

5.2 Use software testing tools.
Assignment 2 15%

Day 12 TUE Sept 29

5.3 Analyze and test product lines.
Activity 5 6%

Day 13 WED Sept 30

Review and Reinforce All Topics
Test 2 20%

Several basic tools and ways of thinking are used in our approach to test management:

- ⋆ In the real world it is impossible to test every single factor that may affect the usability of our software for every purpose it might be applied to.
- ⋆ The current practice in commercial IT development is to use a methodology like agile in developing software, to develop the Traceability Matrix and the use cases, and to identify the core values that your software should deliver. And then test that those value items are being presented in the software. We will refer to the agile best practices guide from the PM Bok (PMI.org) to see how the ideal situation is to prevent problems from occurring by architecting our software properly from the beginning.
- ⋆ A good general principle is that you should always Consider that any piece of code you write may wind up being used in a life-support critical situation Such as controlling the switching for a train track.
- ⋆ As a software professional you are obligated to produce code that is the best you can do.
- ⋆ The biggest problem we encounter is that of edge cases, that is, test cases that occur at the boundary between multiple different sets of inputs.
- ⋆ There is also a psychological dynamic to software quality testing. It is painfully easy to glide over not seeing things that appear to be small because you're focusing on what you think of as larger and more important issues.

Case Study: We will look at the case study of the NASA Mars Climate Orbiter.

`https://blog.cdemi.io/analyzing-software-failure-on-the-nasa-mars-climate-orbiter/`

The elements and dynamics of Software Testing

- ⋆ A solid quality risk analysis. You can't test everything. Therefore, a key challenge to test management is deciding what to test. You need to find the important bugs early in the project. Therefore, a key challenge to test management is sequencing your tests. You sometimes need to drop tests due to schedule pressure. Therefore, a key challenge to test management is test triage in a way that still contains the important risks to system quality. You need to report test results in terms that are meaningful to non-testers. Therefore, a key challenge to test management is tracking and reporting residual levels of risk as test execution continues. Risk based testing, described in this book, will help you do that.
- ⋆ A thorough test plan. A detailed test plan is a crystal ball, allowing you to foresee and prevent potential crises. Such a plan addresses the issues of scope, quality risk management, test strategy, staffing, resources, hardware logistics, configuration management, scheduling, phases, major milestones and phase transitions, and budgeting.
- ⋆ A well-engineered system. Good test systems ferret out, with wicked effectiveness, the bugs that can hurt the product in the market or reduce its acceptance by in-house users. Good test systems mitigate risks to system quality. Good test systems build confidence when the tests finally pass and the bugs get resolved. Good test systems also produce credible, useful, timely information. Good test systems possess internal and external consistency, are easy to learn and use, and build on a set of well-behaved and compatible tools. I use the phrase good test system architecture to characterize such a system. The word architecture fosters a global, structured outlook on test development within the test team. It also conveys to management that creating a good test system involves developing an artifact of elegant construction, with a certain degree of permanence.
- ⋆ A state-based bug tracking database. In the course of testing, you and your intrepid test team will find lots of bugs, a.k.a. issues, defects, errors, problems, faults, and other less-printable descriptions. Trying to keep all these bugs in your head or in a single document courts immediate disaster because you won't be able to communicate effectively within the test team, with programmers, with other development team peers, or with the project management team—and thus won't be able to contribute to increased product quality. You need a way to track each bug through a series of states on its way to closure. I'll show you how to set up and use an effective and simple database that accomplishes this purpose. This database can also summarize the bugs in informative charts that tell management about projected test completion, product stability, system turnaround times, troublesome subsystems, and root causes.

- ⋆ A comprehensive test-tracking spreadsheet. In addition to keeping track of bugs, you need to follow the status of each test case. Does the operating system crash when you use a particular piece of hardware? Does saving a file in a certain format take too long? Which release of the software or hardware failed an important test? A simple set of worksheets in a single spreadsheet can track the results of every single test case, giving you the detail you need to answer these kinds of questions. The detailed worksheets also roll up into summary worksheets that show you the big picture. What percentage of the test cases passed? How many test cases are blocked? How long do the test suites really take to run?

- ⋆ A simple change management database. How many times have you wondered, "How did our schedule get so far out of whack?" Little discrepancies such as slips in hardware or software delivery dates, missing features that block test cases, unavailable test resources, and other seemingly minor changes can hurt. When testing runs late, the whole project slips. You can't prevent test-delaying incidents, but you can keep track of them, which will allow you to bring delays to the attention of your management early and explain the problems effectively. This book presents a simple, efficient database that keeps the crisis of the moment from becoming your next nightmare.

- ⋆ A solid business case for testing. What is the amount of money that testing saves your company? Too few test managers know the answers to this question. However, organizations make tough decisions about the amount of time and effort to invest in any activity based on a cost benefit analysis. We will see how to analyze the testing return on investment, based on solid, well established quality management techniques.